

[Get started](#)[Open in app](#)

## Renato Lelis

38 Followers

[About](#)[Follow](#)

# Deploy de uma aplicação Django no Heroku

[Renato Lelis](#) Aug 7, 2019 · 8 min read

Então sua aplicação está pronta para ver a luz de produção. Testes foram feitos, o cliente deu o OK e agora é só subir em um servidor... Mas como?

## Heroku

[Get started](#)[Open in app](#)

O heroku oferece uma interface com o git e é bem fácil de configurar. Você pode fazer via command line (CLI) e também via interface gráfica que eles disponibilizam. Eu vou mostrar como fazer via CLI (que é o que eu prefiro).

## Configurações

Configurar o ambiente de desenvolvimento localmente pode dar dor de cabeça e o mesmo pode acontecer quando se estar subindo uma aplicação para produção. O importante é fazer um levantamento das configurações de ambiente da sua aplicação e criar um checklist para garantir que tudo foi seguido.

## A aplicação

Para manter o foco, a aplicação que vamos fazer o deploy vai ser bem simples. Vai ser um CRUD básico, de cadastro de contatos sem regras complexas. Você pode acessar o código dela [aqui](#).

## Criando a app no heroku

Como dito anteriormente, o heroku tem uma interface com o git, então naturalmente sua aplicação tem que ser um repositório versionado pelo git. Então comece iniciando dando um git init no seu repositório (caso não tenha feito ainda).

```
git init  
  
git add .  
  
git commit -m"Initial commit"
```

Depois disso, instale o [heroku CLI](#) na sua máquina e crie uma [conta](#) no heroku.

Agora, abra a linha de comando e faça login.

```
heroku login -i
```



[Get started](#)[Open in app](#)

Vá até o diretório da sua aplicação e crie um 'app' no heroku

```
heroku create <nome_app>
```

```
C:\WINDOWS\system32\cmd.exe
C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>heroku create contatos-crud
Creating   contatos-crud... done
https://contatos-crud.herokuapp.com/ | https://git.heroku.com/contatos-crud.git
C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>
```

O nome dessa app tem que ser um nome único no heroku (não deve ser difícil de achar um nome assim).

Antes de fazer o deploy, seu repositório deve ter conter dois arquivos na raiz do diretório da aplicação.

O primeiro deles é o **Procfile**, um arquivo de configuração que contém comandos para serem rodados na inicialização da aplicação. Basta criar o arquivo **Procfile**, sem extensão, e colocar o nome do processo, dois pontos (:) e o comando.

[Get started](#)[Open in app](#)

web que roda em UNIX e serve para vários frameworks web do Python. Vamos usá-lo para rodar o WSGI aplicação do Django.

Primeiro instalamos o **gunicorn** e depois criamos o **Procfile** com o comando para inicializar a aplicação.

```
> pip install gunicorn
```

No **Procfile**:

```
web: gunicorn contatos_base.wsgi
```

O comando web é o nome do processo que nossa aplicação vai rodar. No heroku existem alguns tipos de processos, como nossa aplicação é web então utilizaremos este processo.

O segundo arquivo é o **requirements.txt**, que contém todas as dependências instaladas na sua aplicação (django, gunicorn e etc). Para gerar este arquivo basta rodar o seguinte comando:

```
pip freeze > requirements.txt
```

```
C:\WINDOWS\system32\cmd.exe
(env) C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>pip freeze > requirements.txt
(env) C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>
```

[Get started](#)[Open in app](#)

```
git add requirements.txt Procfile
```

```
git commit -m "Add requirements.txt & Procfile"
```

## Variáveis de ambiente e settings.py

Tendo criado a aplicação no heroku e feito a configuração inicial da aplicação, há algumas outras configurações a nível de código a serem feitas para que a aplicação rode sem problemas.

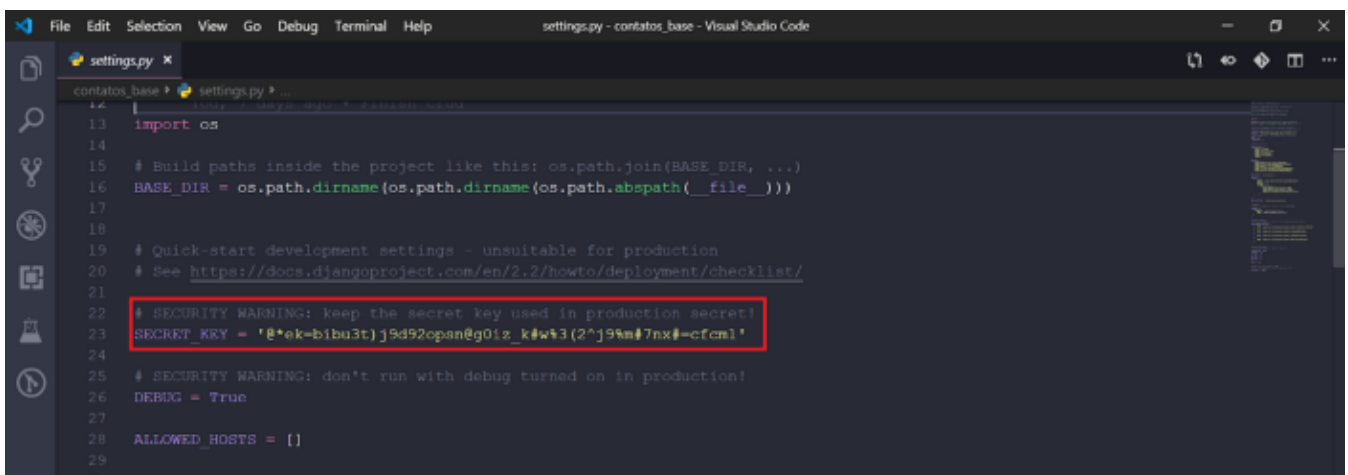
Essas configurações são feitas no arquivo padrão de configuração do Django, o **settings.py**. Essas configurações são de variáveis de ambiente, outras de arquivos estáticos e hosts permitidos. Existe um pacote criado pelo Heroku para facilitar algumas dessas configurações, o django-heroku. Vamo começar instalando ele e depois voltamos para ele.

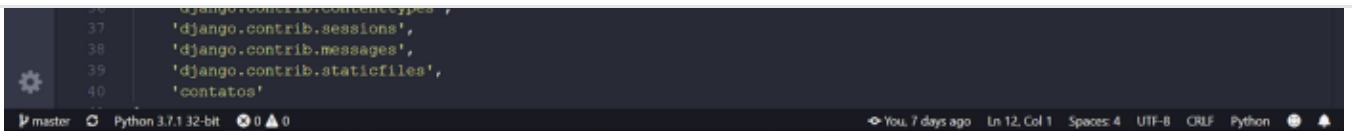
```
pip install django-heroku
```

Não esqueça de adicionar esse pacote no **requirements.txt**

```
pip freeze > requirements.txt
```

A primeira variável que vamos mexer e a **SECRET\_KEY**.



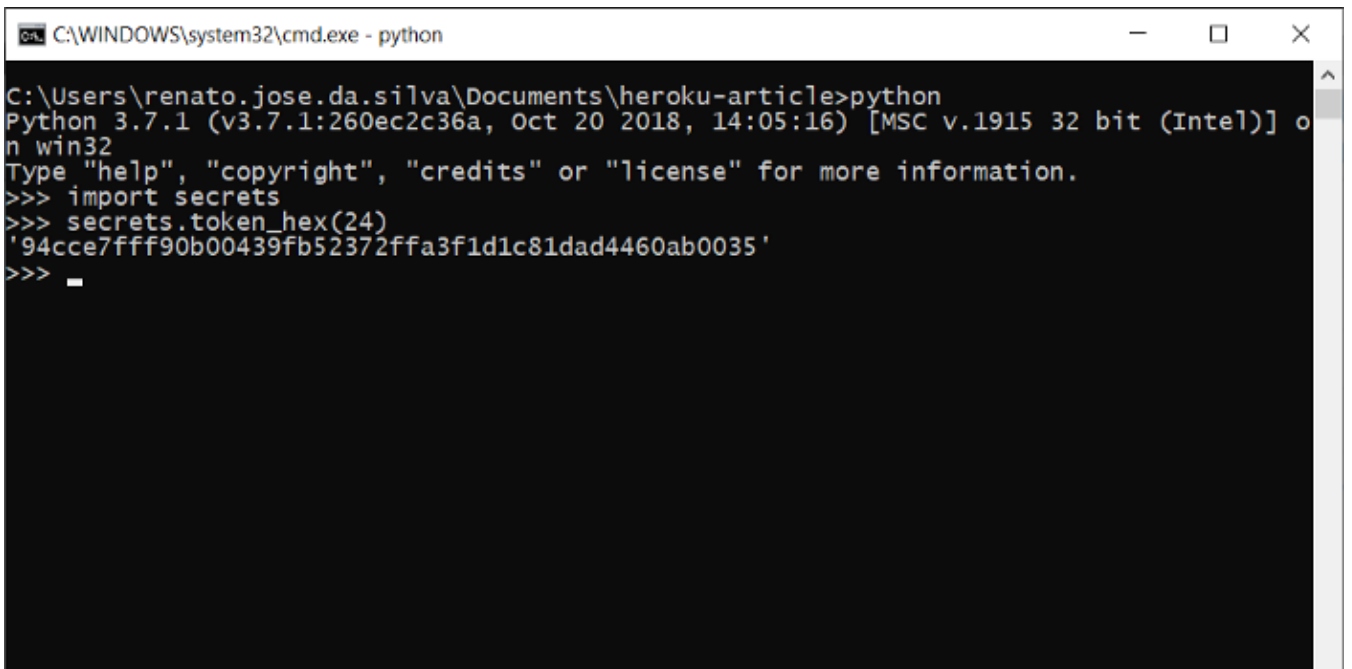
[Get started](#)[Open in app](#)

A **SECRET\_KEY** é usada para criar uma assinatura criptográfica na sua aplicação. Ela deve ter um valor único e imprevisível. Também deve ser mantida em segredo e é claro que a **SECRET\_KEY** que aparece na imagem não vai ser a que eu vou subir para o Heroku ;).

Para gerar uma nova chave, o Python pode ajudar com a biblioteca **secrets**.

Abra o shell do Python com o comando *python*.

```
>>> import secrets  
  
>>> secrets.token_hex(24)
```



A biblioteca **secrets** tem uma função, **token\_hex**, que gera um token com números com base hexadecimal, basta passar o tamanho do token como parâmetro, que é de 24.

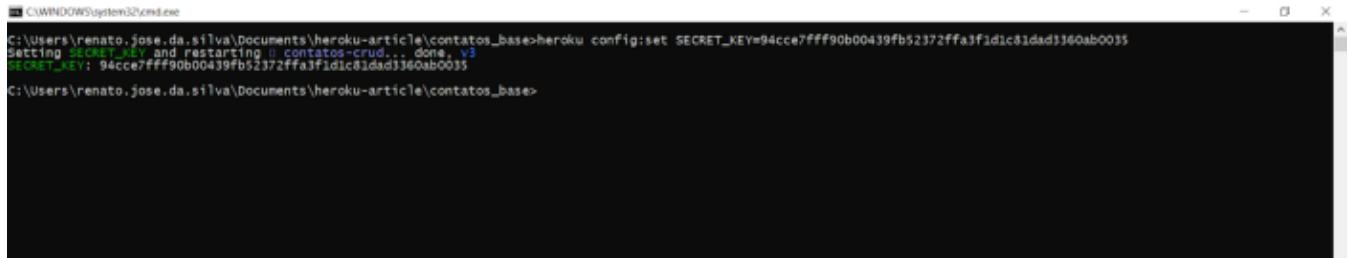
Crie uma variável de ambiente no seu sistema operacional. Se não sabe como fazer isso [aqui](#) tem um tutorial de como fazer no windows e aqui para [Mac/Linux](#). Crie também

Get started

Open in app



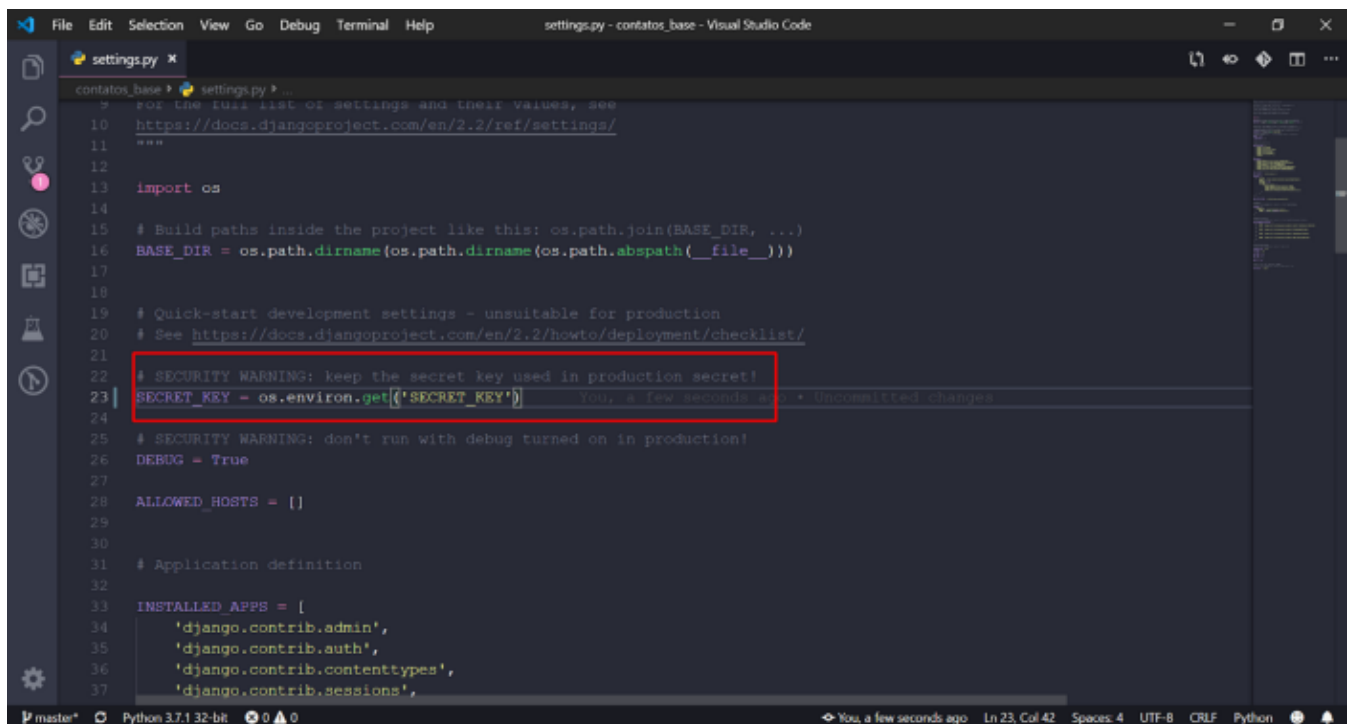
```
heroku config:set SECRET_KEY=CHAVE
```



```
C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>heroku config:set SECRET_KEY=94cce7fff90b00439fb52372ffa3f1d1c81dad3360ab0035
Setting SECRET_KEY and restarting 11 contatos-crud... done, v3
SECRET_KEY: 94cce7fff90b00439fb52372ffa3f1d1c81dad3360ab0035
C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>
```

Com o comando **heroku config:set**, conseguimos criar variáveis de ambiente e adicionar valores a elas. Essas variáveis ficam visíveis em nosso código da seguinte forma:

```
SECRET_KEY = os.environ.get('SECRET_KEY')
```



```
contatos_base > settings.py
# For the full list of settings and their values, see
# https://docs.djangoproject.com/en/2.2/ref/settings/
"""
10
11
12
13 import os
14
15 # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
16 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = os.environ.get('SECRET_KEY')
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38 ]
```

Com o módulo **os** conseguimos pegar o valor da variável **SECRET\_KEY** e qualquer outra variável de ambiente.

Outra variável que devemos transformar em variável de ambiente é a **DEBUG**. Não é uma boa prática essa flag estar ativa em produção mas não é muito prático sempre

[Get started](#)[Open in app](#)

Faça os mesmos passos da **SECRET\_KEY** para criar a variável **DEBUG**.

```
C:\WINDOWS\system32\cmd.exe
C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>heroku config:set DEBUG=False
Setting DEBUG and restarting ⬢ contatos-crud... done, v4
DEBUG: False
C:\Users\renato.jose.da.silva\Documents\heroku-article\contatos_base>_
```

Como toda variável de ambiente é uma String, o valor da variável **DEBUG** vai ser uma condicional.

```
DEBUG = os.environ.get('DEBUG') == True
```

```
File Edit Selection View Go Debug Terminal Help settings.py - contatos_base - Visual Studio Code
contatos_base settings.py
# for the full list of settings and their values, see
# https://docs.djangoproject.com/en/2.2/ref/settings/
"""
10
11
12
13 import os
14
15 # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
16 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = os.environ.get('SECRET_KEY')
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = os.environ.get('DEBUG') == 'True'
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
```

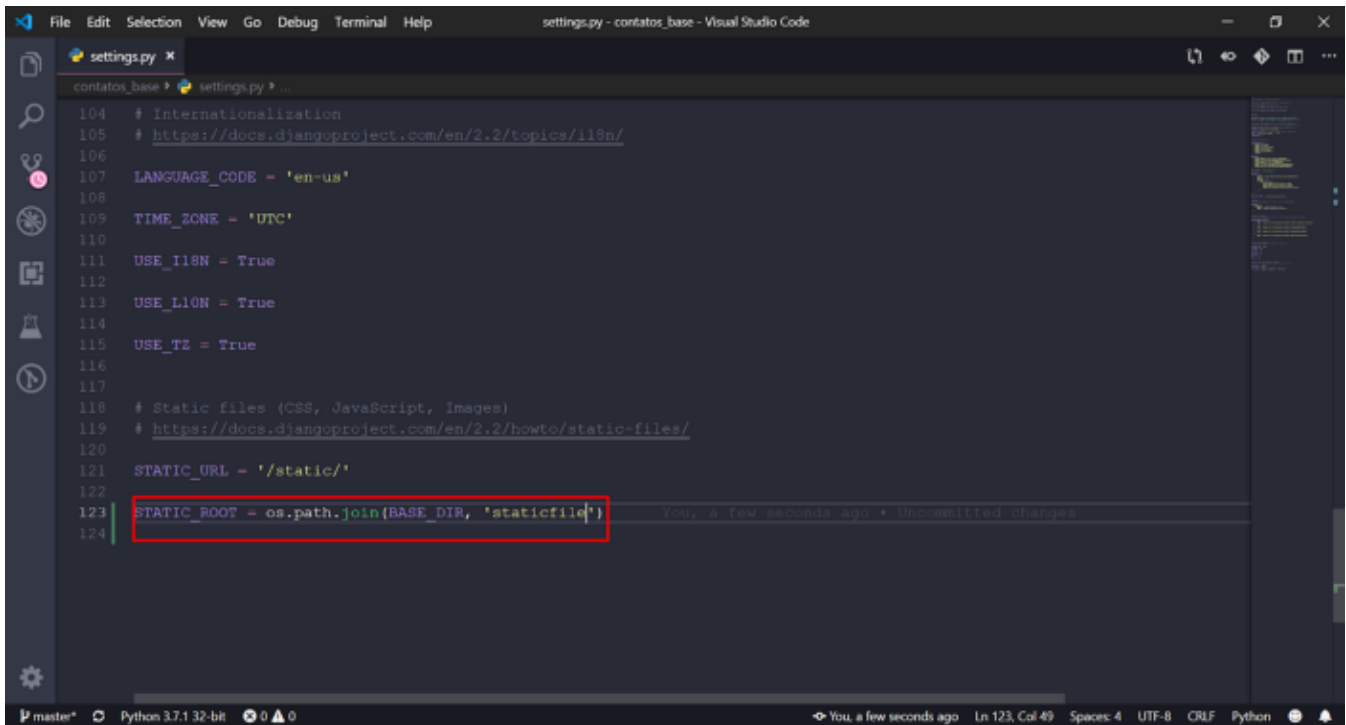
As últimas configurações que vamos fazer no **settings.py** é em relação a arquivos estáticos, host e iremos usar o já instalado **django-heroku** para configurar automaticamente o banco de dados.



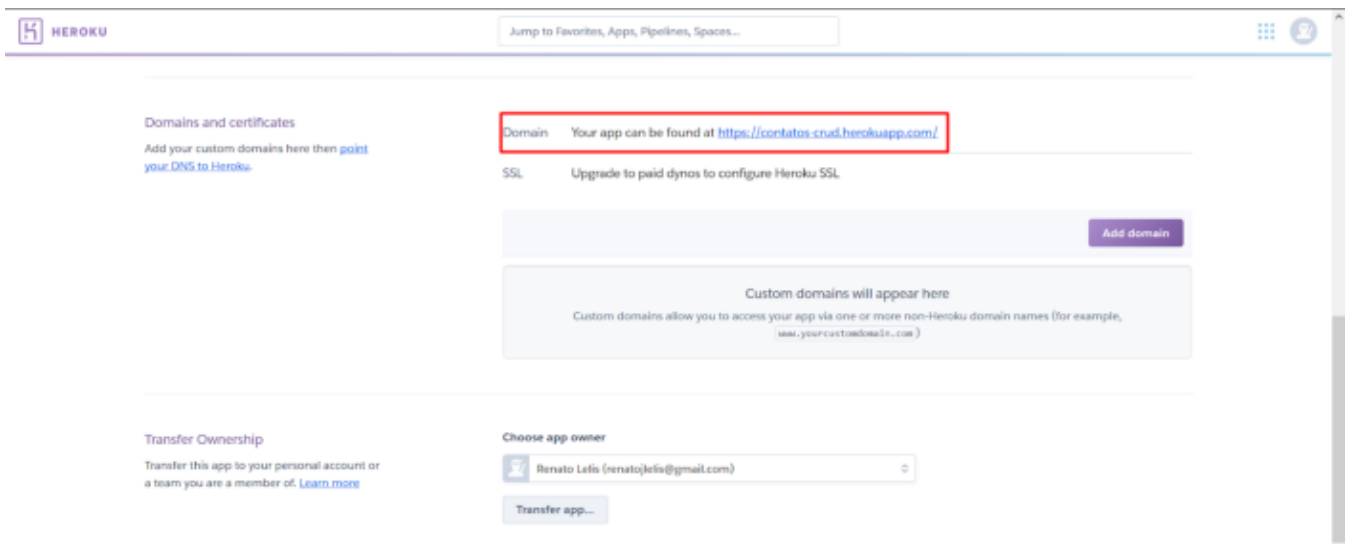
[Get started](#)[Open in app](#)

estáticos em um só diretório. E para saber qual diretório criar usamos a variável **STATIC\_ROOT**.

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfile')
```



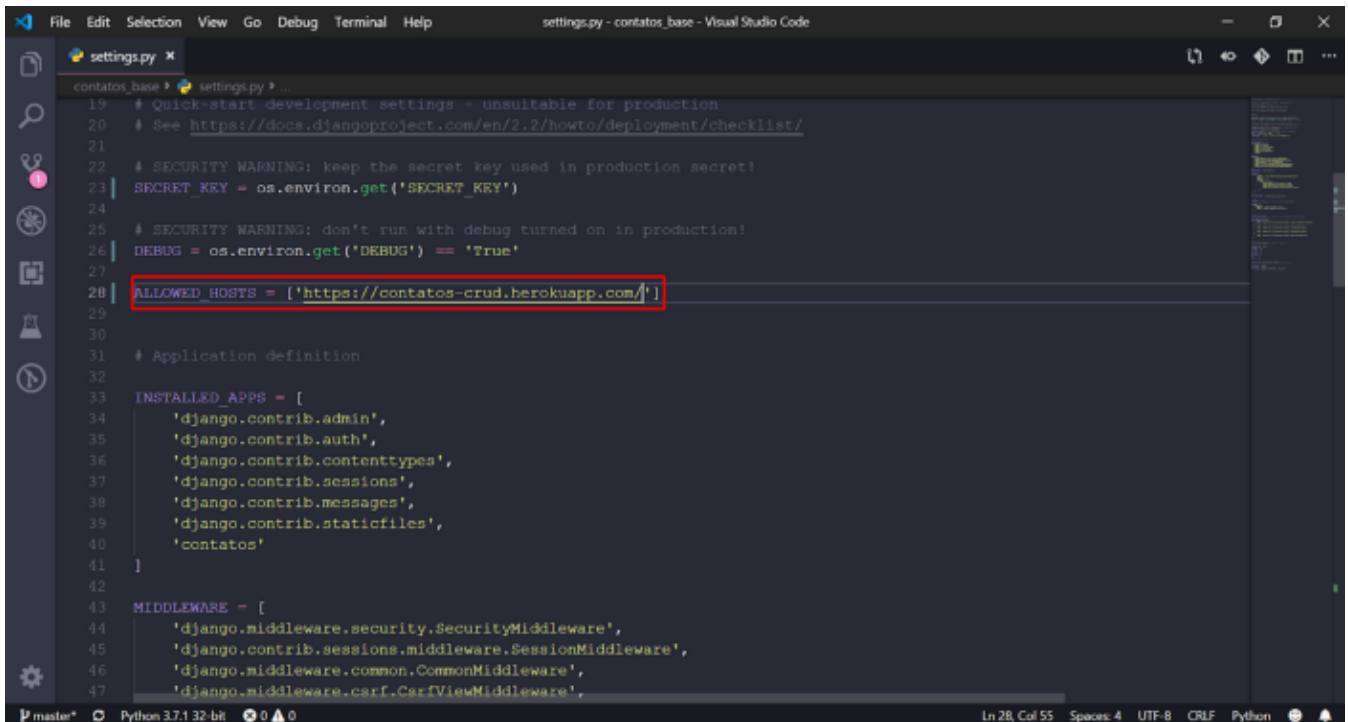
Como medida de segurança, quando **DEBUG** é igual a False, O Django nos força a adicionar os hosts permitidos na nossa aplicação. Isso é bem fácil de fazer, basta adicionar a URL gerada pelo Heroku no momento da criação do app. Se você não sabe a URL, é só entrar no dashboard do Heroku, selecionar a aplicação e clicar em settings.



[Get started](#)[Open in app](#)

Cole o domínio dentro da lista **ALLOWED\_HOSTS**.

```
ALLOWED_HOSTS = ['https://contatos-crud.herokuapp.com']
```



```
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = os.environ.get('SECRET_KEY')
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = os.environ.get('DEBUG') == 'True'
27
28 ALLOWED_HOSTS = ['https://contatos-crud.herokuapp.com/']
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'contatos'
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
```

E a última configuração nesse arquivo é com o `django-heroku`. O `django-heroku` é um módulo criado pelo próprio Heroku para facilitar algumas configurações repetitivas como banco de dados, log, testes e outras coisas. Vamos utilizá-lo para configurar o PostgreSQL como banco de dados da nossa aplicação e é bem simples de usar. Importe ele no topo do `settings.py`.

```
import django_heroku
```

[Get started](#)[Open in app](#)

E final coloque o seguinte comando:

```
django_heroku.settings(locals())
```

Isso vai dar conta de todas as configurações que eu citei mais acima.

## Configurações finais no Heroku

Feito todas essas configurações agora podemos subir nossa aplicação para para o Heroku. Rode o comando:

```
git push heroku master
```

[Get started](#)[Open in app](#)

O Heroku irá instalar tudo necessário para rodar o Django incluindo as dependências contidas no **requirements.txt**.

Localmente estamos usando o SQLite como banco de dados mas no Heroku usaremos o PostgreSQL. Dentro do diretório da aplicação rode o seguinte comando:

```
heroku addons:create heroku-postgresql:hobby-dev
```

Rode as migrations dentro do Heroku e crie um usuário administrador para poder administrar sua aplicação.

```
heroku run python manage.py migrate
```

E para criar um usuário administrador rode o bash dentro do Heroku.

```
heroku run bash
```

E depois:

```
python manage.py createsuperuser
```

[Get started](#)[Open in app](#)

ela finalmente em produção.



Como eu disse no começo, são várias configurações a serem feitas até sua aplicação rodar num servidor sem ser local. Segue um checklist do Django para servir de referência para suas próximas subidas de outras aplicações:

### 1. Logar no heroku via CLI

```
heroku login -i
```

### 2. Iniciar um repositório git

```
git init  
git add .  
git commit -m"Initial commit"
```

### 3. Criar o app no Heroku

[Get started](#)[Open in app](#)

- Executar no diretório do repositório git local

#### 4. Criar o arquivo requirements.txt com todas as dependências

```
pip freeze > requirements.txt
```

- Dependências para uma aplicação Django:
- django
- django-heroku
- gunicorn

#### 5. Criar Procfile

```
web: gunicorn <nome-projeto>.wsgi
```

#### 6. Criar variável STATIC\_ROOT no settings.py

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

#### 7. Colocar variável SECRET\_KEY como variável de ambiente no settings.py

- Gerar a chave em Python

[Get started](#)[Open in app](#)

- Criá-la no **settings.py**

```
SECRET_KEY = os.environ.get('SECRET_KEY')
```

#### 8. Criar variável de ambiente SECRET\_KEY no Heroku

```
heroku config:set SECRET_KEY=<resultado_token_hex>
```

#### 9. Colocar a variável DEBUG do settings.py como variável de ambiente

```
DEBUG = os.environ.get('DEBUG') == 'True'
```

#### 10. Criar variável de ambiente DEBUG no Heroku

```
heroku config:set DEBUG="False"
```

[Get started](#)[Open in app](#)

```
ALLOWED_HOSTS = ["URL"]
```

## 12. Configurar o django-heroku no arquivo settings.py

- No início do arquivo:

```
import django_heroku
```

- No final do arquivo:

```
django_heroku.settings(locals())
```

## 13. Instalar o PostgreSQL no Heroku

```
heroku addons:create heroku-postgresql:hobby-dev
```

## 14. Subir a aplicação no Heroku

```
git add .
```

```
git commit -m"Finish django configurations"
```

```
git push heroku master
```

## 15. Rodar as migrations no Heroku

```
heroku run python manage.py migrate
```

## 16. Criar superuser no Heroku



[Get started](#)[Open in app](#)

```
python manage.py createsuperuser
```

Links úteis:

- [Gunicorn](#)
- [WSGI Servers](#)
- [WSGI](#)
- [Procfile](#)
- [django-heroku](#)
- [Secret key](#)
- [Variaveis de ambiente Windows](#)
- [Variáveis de ambiente Mac/Linux](#)
- [Allowed\\_Host](#)

[Django](#)[Python](#)[Heroku](#)[About](#) [Write](#) [Help](#) [Legal](#)

Get the Medium app

